International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 2 Issue 12, December -2013, pp: (75-80), Available online at: www.erpublications.com

Vector Arithmetic Logic Unit

Amit Kumar Dutta
JIS College of Engineering, Kalyani, WB, India

Abstract: Real time operation of Digital Signal Processing Algorithm for Digital Radio, Digital Television and Data Communication require Giga operations per second. We need Vector Arithmetic Logic Unit to solve that problem. Here we discuss the basic principles of vector addition and vector multiplication and their implementation in logical level.

1.0 INTRODUCTION

A major challenge of today is to design handheld receivers for Software defined Radio or data communication which require Giga operations per second [1]. We think of overcoming those and other calculation intensive scientific problems by using Vector Arithmetic Logic Unit (VALU) which is described here. We found the VALU can solve matrix multiplications and matrix inversions and can solve Fast Fourier Transform (FFT) easily [5].

A vector processor is a Central Processing Unit that implements instructions that operates on one dimensional array of data called vector. Here a logical effort is made to design an adder and multiplier circuit. Using logical method for quick optimization, the parallel and bit serial methods are compared and evaluated. Later stage we choose circuits to make it fastest among other [3][4]. Using logical effort we come to the conclusion that the new design of adder and multiplier is suitable for Vector ALU under a single processor.

Delay estimation and throughput are critical in the development of efficient VLSI processors [2]. Delay estimates are normally presented either in terms of gate delays or processor clock interval. This allows us to properly evaluate different architectures of VLSI processors, specially Arithmetic Logic Unit (ALU) design. The throughput of a VLSI processor determines the rate at which the output of adder or multiplier circuit is generated.

The Section II designs the adder circuit in logical level and a comparison is made between parallel and bit serial method. We extend this idea to multiplier in Section III and design the multiplier which is not an array multiplier, but a bit serial array one. In the Section IV we extend our idea to radix-4 addition and multiplication. In the Section V we explain how Vector ALU is used using two examples of Matrix Multiplication and Matrix Inversion by Gauss-Jordan Method. Section VI explains the real time usage of this VALU in Communication and signal processing domain. This idea can be extended to have a single instruction of matrix multiplication and Inversion of NxN Matrix size by coding hardwired micro instructions.

2.0 ADDER

Here we use single bit adder which is logically as given in the following equation. [3][7].

Sum = x'y'z+x'yz'+xy'z'+xyz
Carry_out = xy+xz+yz
Where, x, y and z(Carry_in) are the inputs and Sum and Carry_out are the outputs.

As we use the single bit adder in bit serial adder structure, we use serial clock which is faster than the parallel clock by N times. Figure 1 shows the simplest bit serial adder, where the carry is fed back to itself through a delay shift register. For 16 bit parallel adder, we require hardware equivalent to more than 16 single bit adders. If we put 16 single bit adders, in bit serial addition we get N times more addition than the equivalent parallel adder. Now we can have two different structures in bit serial form. The first one is the simplest structure, where 16 single bit adders work in parallel, the inputs are bit serial. Figure 2 explains a bit serial adder structure where the all the adders are connected in series, carry is propagated through shift register to the next adder. Here the data is fed one after other to the adder serially and addition takes 16 clock cycles for the first addition and next 15 additions are done in 15 serial clock cycle. Now we look at the matrix operation like loop1: Z(i)=X(i)+Y(i). Both the structures can do this. But the integration operation loop2: Z=Z+X(i) can be done only by

structure shown in Figure 2. So we find that it is Vector addition. We later find that the multiplier is also a Vector operation. Thus we get a Vector Arithmetic Logic Unit.

From Figure 2, we understand the data first is available in 16 [15:0] bit registers and MUXed to the adder bit serially. The registers are MUXed by transmission gates. Similar techniques are used in the output of the adder and they are stored in the register bank. For the integration problem loop2: Z=Z+X(i), we modify the MUX input connection. We take the output of the adder and feed it back to one of the two bit serial inputs of the adder through a shift register. The output of the adder is kept in the output register bank called accumulator.

This can be modified to radix-4 addition.

3.0 MULTIPLIER

The Multiplier-Accumulator (MAC) unit is the main arithmetic processing unit of the Arithmetic Logic Unit (ALU). The multiplier unit is a series connected 16x16 single bit adder which allows 16 multiplications to take place simultaneously. Here N is equal to 16. It accepts up to 2*N input operands and outputs N of 32-bit result of the multiplication stored in the accumulators. The operation of the MAC unit occurs independently and in parallel with adder activity, and its registers facilitate buffering for both Data ALU inputs and outputs. Latches on the MAC unit input permit writing new data to an input register while the Data ALU processes the current data. The input to the multiplier can come only from the X or Y registers. The 32-bit sum is stored back in the same accumulators. The multiply/accumulate operation is fully pipelined and takes 45 serial clock cycles to complete 32 bits output. The multiplication can be between two signed or unsigned integer or fixed point numbers.

The staggered multiplications allow the vector operation Z=sum{X(i)*Y(i)} for i=1 to 16. The Figure 3 shows how we can modify the array multiplier and keep an adder between two shift registers which allows highest clocking frequency. For simplicity, 4x4 bit multiplications are shown in the figure.

4.0 RADIX-4 VECTOR ADDITION and MULTIPLICATION

Vector ALU discussed here needs to be faster and that can be achieved by changing the number system from Radix-2 to Radix-4. The radix-4 has number 0,1,2,3 and addition will be modulo-4 operation. The single bit adder will have a truth table which is simple to find. We find that the CARRY_out is maximum when the inputs are 3,3 and CARRY_in=1, it is 1 and sum is equal to 3. Here the structure of vector ALU adder will remain same except the single bit adder will be different. Logically it is a 2-bit adders (X0, X1 & Y0, Y1 and Carry_in) with two sum output (S0, S1) and a Carry_out.

Vector multiplication in Radix-4 number system has three steps [6]. In the first step, we multiply the multiplicand by the number (0, 1, 2, 3) (multiplier) using a look up table to produce a partial product which is implemented in hardware. In the second step, we convert the partial sum to radix-4 by modulo-4 operation. In the third step, we add partial sums shifted by two bits each time. Then we add every two partial sums.

5.0 MATRIX MULTIPLICATION and INVERSION

Vector ALU is used mainly for mathematical equation solving. In Signal Processing, Image Processing and Control System Application, matrix multiplication and inversion are mainly used. In Vector ALU, staggered Adders and Multipliers are used to solve those equations. Here we check the time required for two 16x16 matrix multiplication.

$$Z = \begin{bmatrix} X_{1,1} & X_{1,2} \dots\dots & X_{1,16} \\ X_{2,1} & X_{2,2} \dots\dots & X_{2,16} \\ X_{16,1} & X_{16,2} \dots\dots & X_{16,16} \end{bmatrix} \begin{bmatrix} Y_{1,1} & Y_{1,2} \dots\dots & Y_{1,16} \\ Y_{2,1} & Y_{2,2} \dots\dots & Y_{2,16} \\ Y_{16,1} & Y_{16,2} \dots\dots & Y_{16,16} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} \dots\dots & Z_{1,16} \\ Z_{2,1} & Z_{2,2} \dots\dots & Z_{2,16} \\ Z_{16,1} & Z_{16,2} \dots\dots & Z_{16,16} \end{bmatrix}$$

Where

$$Z_{i,j} = \sum_{n=1}^{16} X_{i,n} Y_{n,j}$$

It has 16x16x16 multiplications and 15x16x16 additions in one matrix multiplication. Because, 16 operations are done in parallel, we roughly need 1 millisecond for the matrix multiplication.
In matrix inversion, we use Gauss-Jordan method where we try to minimize the use of division. Assume the matrix to be X

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} \ldots\ldots & X_{1,16} & 1 & 0 \ldots\ldots & 0 \\ X_{2,1} & X_{2,2} \ldots\ldots & X_{2,16} & 0 & 1 \ldots\ldots & 0 \\ X_{16,1} & X_{16,2} \ldots\ldots & X_{16,16} & 0 & 0 \ldots\ldots & 1 \end{bmatrix}$$

We divide the row1 by $X_{1,1}$. We implement it by taking $1/X_{1,1}$ and then multiplying the row by that number. Then we multiply the row by $X_{2,1}$ and subtract it from row2 and similarly from all the rows. This way the algorithm is implemented. It also requires 1 millisecond time.

We can implement Fast Fourier Transform (FFT) algorithm using Vector ALU for conversion from time domain to frequency domain.

6.0 USAGE of VALU

We can implement various communication algorithm and signal processing using VALU [1][6]. For example, we can use VALU for Decision Feedback Canceller for QAM-WCDMA transmission, FFT/IFFT block in OFDM, Code Excited linear predictive coding for compressing and digitizing voice signal and different filters for signal processing. In QAM-WCDMA transmission we can increase capacity ten times than FDMA by using shifted m-sequence with adaptive receiver. In line communication the effect of fading is minimum and the adaptive receiver has to solve the Wiener-Hopf equation only occasionally which is given by $\mathbf{w}=\mathbf{R}^{-1}\mathbf{p}$. The correlation matrix **R** is known and for a chip length of 31, it can be easily found. **p** is the shifted m-sequence. We can find that in **w** the noise variance is reduced by a factor of 16. So we can compress the signal in time domain more to have the same noise variance in the receiver. There are other methods to achieve more users.

For mobile receiver, the fading is critical and the matrix **R** is evaluated from the RMS delay spread measured at the hand set at an interval of predetermined time. The adapted coefficients **w** are calculated every time and VALU can solve this problem.

In OFDM we need to take FFT in the receiver and one can implement the butterfly structure in VALU. The linear predictive coding of speech has to find the correlation matrix and its inverse. The filters take convolution. These can be put in matrix form and be solved by VALU.

7.0 CONCLUSION

Addition and multiplication are the main operations of Arithmetic Logic Unit. We improved the speed of addition by a factor of serial clock/parallel clock and the throughput by same amount. This improvement can also be achieved by bit serial adders operating in parallel, but vector integration is not possible. In multiplication we put 16x9 adders for 8 16bit multipliers. So the throughput is eight times and it takes 65 serial clocks for whole operation.

REFERENCES

[1] K. V. Berkel et al., "Vector Processing as an enabler for Software-Defined Radio in Handheld Devices," EURASIP Journal on Applied Signal Processing, 2005.
[2] H. Q. Dao & V. G. Oklobdzija, "Performance Comparison of VLSI adders using logical efforts," B Hachet et al. (Eds) Patmos 2002 INC.
[3] M. Ozawa et al., "Performance evaluation of cascade ALU architecture for asynchronous super-scalar processor," ASYNC 2001.
[4] A. Khalir et al., "High Speed Multiple valued logic full adder using Carbon Nano-tube Field Effect Transistor," International Journal of VLSI design & Communication Systems, Vol. 2, No. 1, March 2011.
[5] Araki T. et al., "The architecture of a Vector digital Signal Processor for video Coding," IEEE International Conference on Accoustic Speech & Signal Processing 1992, ICASSP.
[6] B. Parhami, "Computer Architecture-from Microprocessor to Supercomputer," Oxford Series, 2005.
[7] M. M. Mano, M. D. Ciletti, "Digital Design," Prentice Hall, 4th Edn.

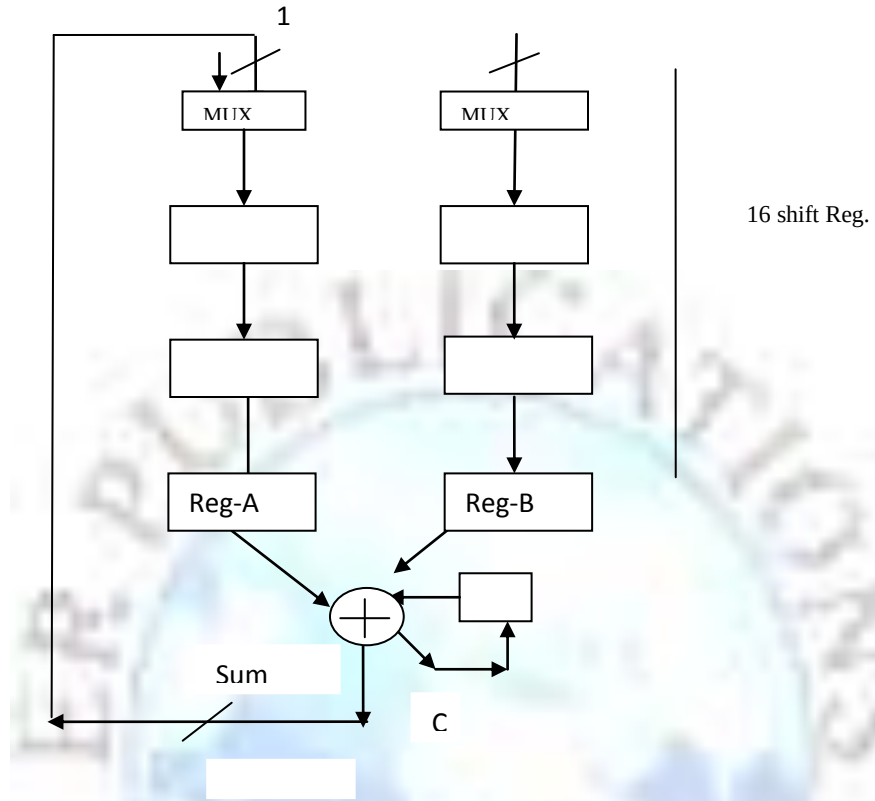

Figure 1. Bit serial Data path for adder.

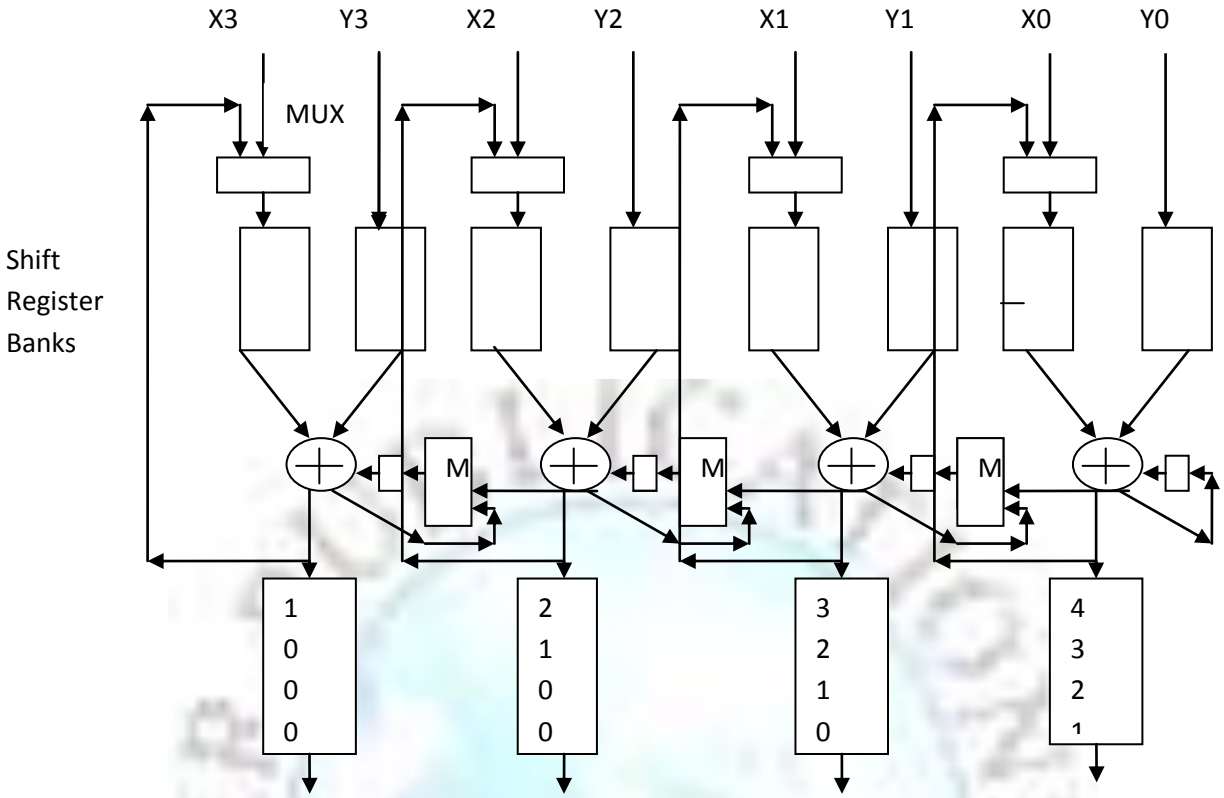

Figure 2. Proposed 4 bit Vector Data Path for Addition. The data is serially passed through shift registers. MUX can replace the shift register banks.

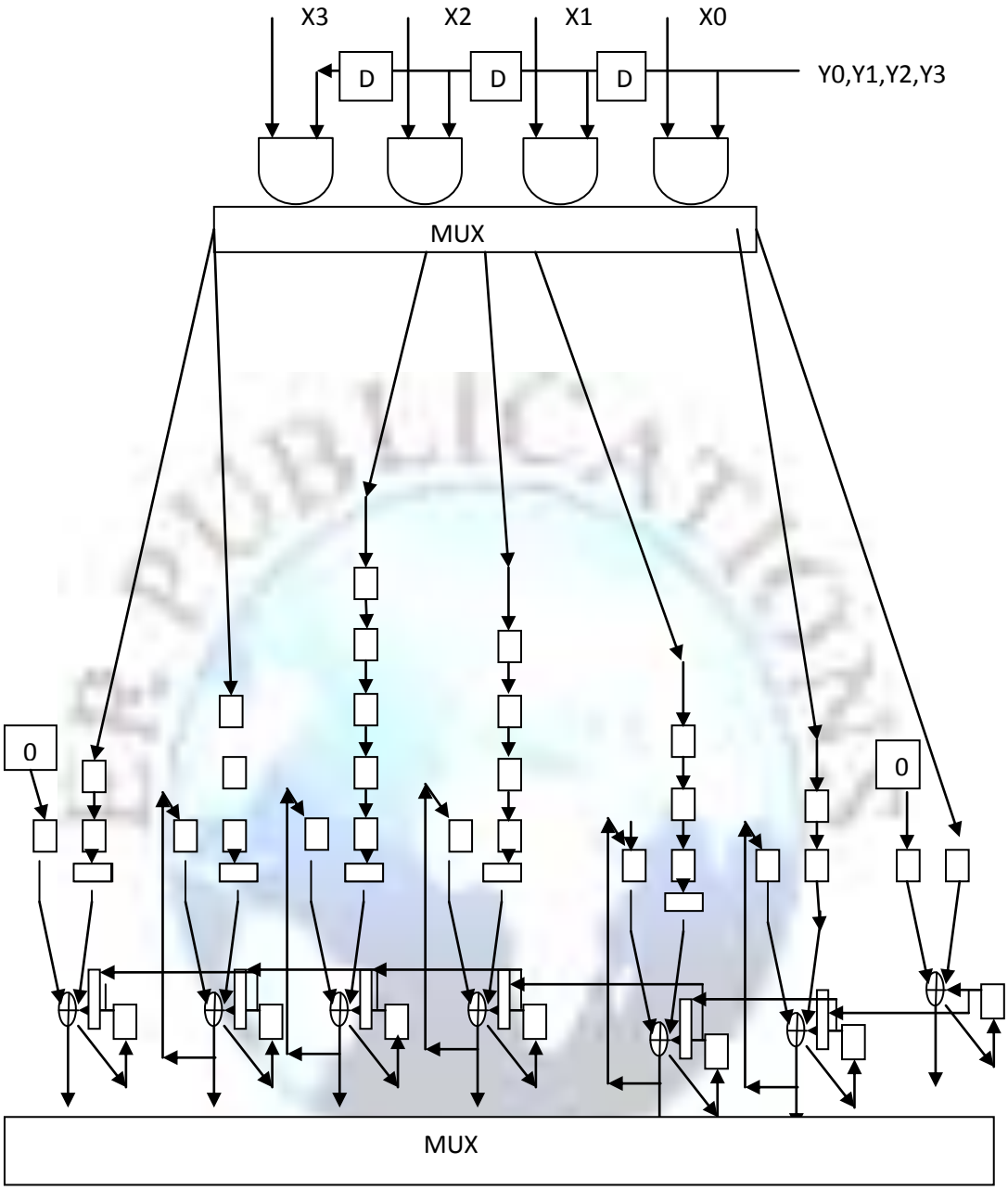

Figure 3. 4x4 bit Vector Multiplier. We need to add 3 more stages of 4 adders to get 4 multipliers.